IFW

PTO/SB/21 (08-00)
Approved for use through 10/31/02. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMENRCE
Under the Paper Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays avalid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/626,121	
Filing Date	July 23, 2003	
First Named Inventor	Pedro Vazquez	
Group Art Unit	~~NYA~~ 2142	
Examiner Name	NYA	
Total Number of Pages in This Submission: 27	Attorney Docket Number	15437-0630

ENCLOSURES *(check all that apply)*

- [ ] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment / Response
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [x] Certified Copy of Priority Document(s)
  from France dtd 30 Avr. 2003
- [ ] Response to Missing Parts/ Incomplete Application
  - [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Assignment Papers *(for an Application)*
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition To Convert To a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, number of CD(s)_______
- [ ] After Allowance Communication to Group
- [ ] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- [ ] Proprietary Information
- [ ] Status Letter
- [ ] Other Enclosure(s) *(please identify below)*:

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Hickman Palermo Truong & Becker LLP Craig G. Holmes
Signature	[signature]
Date	June 3, 2004

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class: mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this date: 6/3/04

Type or printed name	Annette Jacobs		
Signature	[signature]	Date	June 3, 2004

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time your are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

P7666

THIS PAGE BLANK (USPTO)

02.09845
Lettre (1)

R E P U B L I Q U E   F R A N Ç A I S E

INPI
INSTITUT
NATIONAL DE
LA PROPRIETE
INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 30 AVR. 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brévets

Martine PLANCHE

INSTITUT
NATIONAL DE
LA PROPRIETE
INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr

ETABLISSEMENT PUBLIC NATIONAL CREE PAR LA LOI N° 51-444 DU 19 AVRIL 1951

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

cerfa
N° 11354*01

Code de la propriété intellectuelle - Livre VI

REQUÊTE EN DÉLIVRANCE 1/2

Important ! Remplir impérativement la 2ème page.

Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 W / 190600

Réservé à l'INPI REMISE DES PIÈCES DATE 23 JUIL 2002 LIEU 75 INPI PARIS N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0209345 DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 23 JUIL. 2002	**1** NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE CABINET NETTER 36 avenue Hoche 75008 PARIS
Vos références pour ce dossier *(facultatif)* SUN 41 (120724)	

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	**Cochez l'une des 4 cases suivantes**
Demande de brevet	☒
Demande de certificat d'utilité	☐
Demande divisionnaire	☐
Demande de brevet initiale	N° Date / /
ou demande de certificat d'utilité initiale	N° Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N° Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Server and API for distributed rendezvous.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	Pays ou organisation Date / / N° Pays ou organisation Date / / N° Pays ou organisation Date / / N° ☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**

5 DEMANDEUR		☐ **S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»**
Nom ou dénomination sociale		SUN MICROSYSTEMS, INC
Prénoms		
Forme juridique		
N° SIREN		
Code APE-NAF		
Adresse	Rue	901 San Antonio Road
	Code postal et ville	94303 PALO ALTO Californie
Pays		Etats-Unis d'Amérique
Nationalité		Société des Etats-Unis d'Amérique
N° de téléphone *(facultatif)*		
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI	
REMISE DES PIÈCES DATE	23 JUIL 2002
LIEU	75 INPI PARIS
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI	0209345

DB 540 W /190600

Vos références pour ce dossier : *(facultatif)*		SUN Aff. 41 (120724)
6 MANDATAIRE		
Nom		PLAÇAIS
Prénom		Jean-Yves
Cabinet ou Société		Cabinet NETTER
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	36 avenue Hoche
	Code postal et ville	75008 PARIS
N° de téléphone *(facultatif)*		01 58 36 44 22
N° de télécopie *(facultatif)*		01 42 25 00 45
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		
Les inventeurs sont les demandeurs		☐ Oui ☒ Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☐ ☒
Paiement échelonné de la redevance		**Paiement en deux versements, uniquement pour les personnes physiques** ☐ Oui ☐ Non
9 RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence) :*
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire) N° Conseil 92-1197 (B) (M) Jean-Yves PLAÇAIS	**VISA DE LA PRÉFECTURE OU DE L'INPI** L. MARIELLO

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

Server and API for distributed rendezvous

The invention relates to a distributed computer system, for example a distributed computer system providing a distributed software execution environment.

Such a system or a platform is composed of a group of cooperating computers, called e.g. nodes. Each node has programs to be executed, a program being a set of code instructions; A program in execution may be designated as a process.

In a computer, a process can adapt its execution according to the execution of other processes in the same computer. A process may require to adapt its execution to other process executions for different reasons : for example, a process may have to wait for another process to finish a task, several processes may have to begin their execution at the same time (start synchronization); several processes may have to wait for each other to execute a code instruction.

In a distributed computer system, such execution adaptation problems are enhanced as nodes do not share memory and communicate only through a network.

The present invention provides advances for synchronization between processes.

The invention concerns a management module comprising :
- input code, capable of receiving a synchronization request message, such a synchronization request message comprising a node meeting identifier and a number of participants,
- table code, capable of maintaining a record comprising node meeting identifier and number of participants for each received synchronization request message,
- parsing code, capable of, responsive to a received synchronization request message, checking whether the number of records having the same node meeting identifier and the number of participants comprised in a record having that node meeting identifier meets a given condition,
- output code, responsive to a record meeting the given condition, capable of transmitting a synchronization-attained message.

α

The invention also concerns a method for managing code execution synchronization, said method comprising the steps of:

a. receiving a synchronization request message, such a synchronization request message comprising a node meeting identifier and a number of participants,

b. maintaining a record of node meeting identifier and number of participants for each received synchronization request message,

c. responsive to a received synchronization request message, checking whether the number of records having the same node meeting identifier and the number of participants in a record having that node meeting identifier meet a given condition,

d. responsive to a record meeting the given condition, transmitting a synchronization-attained message.

Other alternative features and advantages of the invention will appear in the detailed description below and in the appended drawings, in which :

- figure 1 is a general diagram of a computer machine;

- figure 2 is a general diagram of a distributed computer system;

- figure 3 is the general diagram of a distributed computer of figure 2 comprising a rendezvous server according to the invention;

- figure 4 is partial diagram of figure 3 indicating the synchronization messages between a computer machine and the rendezvous server;

- figure 5 illustrates a flow-chart of management of received synchronization messages in the rendezvous server according to the invention.

- figure 6 illustrates a flow-chart of evaluation of a sub-list in the rendezvous server according to the invention.

Trademark Office patent file or records, but otherwise reserves all copyright and/or author's rights whatsoever.

These drawings are placed apart for the purpose of clarifying the detailed description, and of enabling easier reference. They nevertheless form an integral part of the description of the present invention.

Embodiments of this invention may be implemented in a network comprising computer systems. The hardware of such computer systems is for example as shown in Fig. 1, where in the computer system M:

- 1 is a processor, e.g. an Ultra-Sparc processor (SPARC is a Trademark of SPARC International Inc);
- 2 is a program memory, e.g. an EPROM for BIOS;
- 3 is a working memory, e.g. a RAM of any suitable technology (SDRAM for example); and
- 7 is a network interface device connected to a communication medium 9, itself in communication with other computers. Network interface device 7 may be an Ethernet device, a serial line device, or an ATM device, inter alia. Medium 9 may be based on wire cables, fiber optics, or radio-communications, for example.

Data may be exchanged between the components of Figure 1 through a bus system 8, schematically shown as a single bus for simplification of the drawing. As is known, bus systems may often include a processor bus, e.g. of the PCI type, connected via appropriate bridges to e.g. an ISA bus and/or an SCSI bus.

The computer system M may be a node amongst a group of nodes in a distributed computer system.

Figure 2 represents a distributed computer system comprising different computer systems as M1, M2, M3, M4, also called nodes. Each computer system is linked to the other computer systems via the communication medium 9 which may be e.g. a medium based on ethernet. As seen, each of these computer systems comprises hardware as described in figure 1. For example, the hardware of computer system M1 is composed of processor 1-M1, a

α

memory (program or working memory) 4-M1 shared for the processes executed in the computer system M1. When different processes are executed in this computer system, a process can adapt its execution responsive to the execution of one or more processes using the shared memory 4-M1.

Each node has an application layer comprising different processes being programs (code instructions) in execution. For example, process execution adaptation may mean a synchronization between execution starts (or restarts after a wait, e.g. an interruption) between two or more processes. Thus a process may have to wait for another process to finish a task, several processes may have to begin their execution at the same time, several processes may have to wait for each other when executing some code instructions. The synchronization requires for example a process to stop its execution at a waiting instruction and to wait for other processes until their execution arrives to a given instruction: the shared memory enables the process to have information about the execution of other processes and to wait until other processes reach a given code instruction in their execution before start (or restart) its execution.

Processes of nodes may need to adapt their execution with the execution of other processes of other nodes in the distributed computer system. In a distributed computer system, problems are enhanced as nodes do not share memory and communicate only through a network.

The invention proposes a solution to this problem.

Figure 3 illustrates a distributed computer system according to the invention. The distributed computer system comprises a group of computer systems, e.g. the computer systems M1, M2, M3, M4, linked together. Computer systems in the group are linked to a synchronization server 20 called RV-server through the network 9. The computer systems (or nodes) may also be called clients in the client-server environment. The synchronization server is adapted to centralize synchronization requests of computer systems. As the other computer systems of the distributed computer system, computer system M1 comprises software layers such as
* application layer 12-M1,

* process layer 14-M1, comprising programs which may be in execution and thus define process or processes of the application layer,

* RV-API layer 15-M1.

The RV-API provides functions to enables the computer system to communicate with the RV-server 20. A process in the computer system is adapted to call a synchronization request function in the RV-API, thus this process may be called a calling process. The synchronization request function is adapted to establish a link between the computer system and the RV-server, to control input parameters, to send synchronization request messages, to receive and to interpret received information messages e.g. from the RV-server.

More specifically, the execution adaptation between some processes of various computers requires to define a code instruction in these processes for a synchronization request function call. This code instruction indicates a "rendezvous" indication being an indication of a given execution adaptation between some processes, this "rendezvous" indication being e.g. similar for the processes requiring an execution adaptation between them. A rendezvous indication may also be designated as a node meeting indication. The code instruction also requires the process to call the synchronization request function for sending a synchronization request message to the RV-server. This synchronization request message defines parameters for the process such as a given number of participants. This given number of participants represents the number of processes participating to the "rendezvous" awaited by the calling process. The condition for a "reached rendezvous" for the calling process is when the number of received synchronization request messages in the server, indicating the corresponding process is a participant to the rendezvous, is equal to or greater than the number of participants awaited by the calling process. Its execution may continue.

As illustrated in exhibit A-1, a process 1 stops its execution at the RV1 code instruction, calls a synchronization request function to send a synchronization request message to the RV-server, waits for, for example, another synchronization request message, indicating the corresponding process is a participant to the rendezvous RV1, so that process 1 can reach the rendezvous RV1. A process 2 also stops its code execution at the RV1 code instruction, calls a synchronization request function to send a synchronization request message to the RV-server. The synchronization request message indicates that the process 2 requests another synchronization request message, indicating the corresponding process is a

participant to the rendezvous RV1, to reach the rendezvous RV1. Once both synchronization request messages, indicating the respective process 1 and 2 are participant to the rendezvous RV1, are in the server, both processes 1 and 2 reach the rendezvous RV1. Their execution may continue. The cadence of execution of process 2 may be greater than the cadence of execution of process 1: in this case, process 1 stops its execution at the RV1 code instruction before process 2.

A synchronization request function of the API layer 15 is developed in exhibits A-2. This function has different entry parameters to define the RV-server and a given adaptation of executions ("rendezvous") between some processes :

* *rv_id* : this parameter identifies a given adaptation of executions between at least two processes ("rendezvous"). Different processes, on different systems, may meet on a particular rendezvous indicated through this identification.

* *rv_total_wait_num* : this parameter is the total number of participants awaited by the calling process for the rv_id rendezvous. In an embodiment, the calling process is counted in this total number, this process being a participant to the rendezvous.

* *rv_time_out* : this parameter is the maximal time that the calling process waits to complete the rendezvous. The unit of this parameter may be the second. In an embodiment, a value of zero may indicate the calling process waits for ever.

* *rv_block* : this parameter indicates if the calling process is blocked until a rendez vous condition is completed or the timeout expires, or if the calling process is not blocked. For example, a value different from 0 indicates a blocking rendezvous meaning the process interrupt its execution until the rendezvous is attained. On the contrary, a value equal to 0 indicates the calling process is not blocked when this call is executed. In this last case, the process continues its execution does not wait for the execution of other processes.

* *participate* : this parameter indicates if the calling process participates to the rendezvous or not. For example, a value different from 0 indicates the calling process participates to the rendezvous. On the contrary, a value equal to 0 indicates the calling process does not participate to the rende-vous.

* *host_name* : this parameter indicates the name of the host where the rendezvous server is.

* *port* : this parameter indicates the port where the rendezvous server is listening.

α

The operation of the RV-server 20 in relation with a computer system M is illustrated in more details in figure 4. Thus, a process of the computer system M uses its RV- API 15-M and its synchronization request function 13-M to establish a communication with the RV-server. The synchronization request function is arranged to send a connection request to the RV-server, using e.g. the TCP/IP socket from the RV-API, this connection request indicating for example the *host_name* parameter and the *port* parameter of the RV-server. Responsive to this connection request, the RV-server is adapted to establish a connection, e.g. in creating and designating a socket on the RV-server, using TCP/IP socket. This connection may be maintained until communication between the process and the RV-server is over. In another embodiment, the synchronization request function may not maintain this connection and may indicate the RV-server address and the port identifier in each message to send to the RV-server, these RV-server address and the port identifier being parameters provided by the calling process.

Then, the process of the computer system M uses its RV- API 15-M and its synchronization request function 13-M to send a synchronization request message to the RV-server through the connection. As illustrated in Exhibit B-1, this synchronization request message comprises various fields which are at least some of the parameters of the synchronization request function. For example these fields may be *RV-ID*, *RV-total-wait-num*, *participate*, *RV-Block*. In an embodiment of the invention, the called synchronization request function sends a synchronization request message through the connection between the computer of the calling process and the RV-server. The synchronization request message may indicate either the process is not a participant because of e.g. execution problems, either the process is a participant to the rendezvous. By sending a synchronization request message with a non participate indication to the server, the RV-server can release other calling processes blocked and waiting for a given number of calling processes to continue their code execution.

The RV-server creates different sockets; a given socket may be attributed to a connection between the RV-server and the computer of the calling process. The different sockets are input/output adapted to receive the messages and to return messages as described hereinafter. The RV-server 20 also comprises a RV-management module 22 comprising
- input code 22-1 to receive a synchronization request message,

ɑ

- table code 22-3 to maintain a list 24 of records, each record comprising some of the fields of a received synchronization request message such as a rendezvous identifier and a number of participants, the table code being arranged to count the number of records having the same rendezvous identifier and to count the number of records having the same rendezvous identifier and indicating the process as a participant to the rendezvous,
- parsing code 22-4 to check, responsive to a new received synchronization request message, whether the number of records having the same rendezvous identifier is equal to or greater than *RV-total-wait-num* being the number of participants indicated in a record having that rendezvous identifier,
- output code 22-2, responsive to a record meeting this condition, to transmit a synchronization-attained message to the calling process, this synchronization-attained message comprising the number of records having the same rendezvous and indicating the calling process as a participant to the rendezvous.

The RV-management module 22 is adapted to create sub-lists in the list 24. These sub-lists are created to store received synchronization request messages having the designation of the same rendezvous. Thus, sub-list 24-1 is adapted to store synchronization request messages designating the RV-ID1 rendezvous, sub-list 24-n is adapted to store synchronization request messages designating the RV-IDn rendezvous.

The synchronization request function 13 -M is a part of the machine M by way of example only. The synchronization request function 13-M may also be external to the machine M and disposable on the network.

Figure 5 illustrates the flow-chart describing the functions of the management module of the RV-server. At operation 102, the RV-server receives the synchronization request message from the synchronization request function called by a calling process of a node. In this message, the RV-server checks if the identification of the RV-ID rendezvous is new or if it already exists a sub-list dedicated to records having this RV-ID (operation 104). If this identification of the RV-ID rendezvous is new, the RV-server creates a new sub-list, dedicates this sub-list to records having the same identification of the RV-ID rendezvous (operation 105) and records some fields of the synchronization request message (operation 106). If this identification of the RV-ID rendezvous already exists, the RV-server adds the

α

fields of the synchronization request message to the existing sub-list dedicated to records having the same identification of the RV-ID rendezvous (operation 106). The server counts the total number of records in each sub-list and, in a sub-list, the number of records indicating the process as a participant to the rendezvous. For example, this number of records having a participation indication is evaluated by counting the number of records in the sub-list whose *participate* field indicates another value than the value 0. After operation 106, the RV-server is adapted to evaluate the sub-list (operation 108). The flow-chart restarts at operation 102 if a new synchronization request message is received.

Different cases are to be considered to evaluate the sub-list at operation 108. These cases are described in the flow-chart of figure 6. At operation 602, the RV-management retrieves the information of the first record in the sub-list. When the *RV-total-wait-num* of the record is strictly greater than the total number of records in the sub-list at operation 604, the flow-chart continues at operation 612. If the total number of records in the sub-list is equal or greater than the *RV-total-wait-num* of the record at operation 604, the RV-management checks if the *RV-block* field in the record indicates that the process is blocked at operation 606. If it is not, the synchronization request function may return the value 0 as the process is already unblocked, meaning it does not wait for a particular rendezvous to continue its execution; the RV-management deletes this record in the sub-list as the calling process does not wait for any message at operation 610. Else, a synchronization-attained message is transmitted to the calling process. More precisely, this synchronization-attained message is transmitted to the called synchronization request function at operation 608 corresponding to this record. The synchronization request function returns this synchronization-attained message to the calling process. Thus, the synchronization-attained message (which is the return of the synchronization request function) may indicate to the calling process for example this total number of records. The synchronization-attained message may advantageously indicate to the calling process the number of records in the sub-list having a participation indication. The synchronization-attained message specifying the number of records having a participation indication enables the process, which has been blocked to wait for the rendezvous, to take a decision knowing this number of participants and the number of participants awaited by the process. The record of the sub-list is then deleted at operation 610.

If it exists a next record in the sub-list at operation 612, the flow-chart continues for this next record at operation 602, else, the flow-chart ends. The evaluation of the sub-list may be processed when a synchronization request message is added in the sub-list.

The synchronization request function may also return other indications as an error indication. By way of example, the value -1 indicates an error.

Before establishing a connection with the RV-server and sending the synchronization request message to it, the synchronization request function may do some controls as e.g. parameters controls (their value are controlled to be positive). If some controls reveal a problem, e.g. responsive to a detected error on a parameter, an error message is sent from the synchronization request function to the calling process. The error message may be understand as the return of the synchronization request function.

The error message may comprise another number, e.g. the *errno* known in C language (as indicated in "The C programming language", Brian W. Kernighan, Dennis Ritchie, March 1989), which may be print and indicates to the user the type of error. In the example of the implementation of the invention, other types of error may be specified as :

* a bad identification error (number equal to 301) and/or
* a bad number error (number equal to 302) and/or
* a time-out error (number equal to 304) and/or
* other errors as an error concerning the set of an alarm flag (number equal to 303).

Thus, the synchronization request function may launch a clock which counts until the time-out value is reached. If the time-out is reached before a synchronization-attained message is received from the server, the synchronization request function may return to the process an error message specifying the *errno* number for the time-out error. The error message comprises the value (-1) and the *errno* number (304).

The Exhibit B-2, B-3, B-4 illustrate a sub-list at different times T1, T2, T3. For simplification reasons, T1<T2<T3. Moreover, the three synchronization request messages in this example contain fields *participate* and *RV-block* set to 1.

α

In Exhibit B-2, at time T1, the RV-server receives a synchronization request message from a synchronization request function called by a calling process of a computer system M1. As RV3 is a new RV-ID of a meeting point, the RV-server creates the sub-list and records some fields of this synchronization request message. In this synchronization request message, *RV-total-wait-num* is equal to 3.

In Exhibit B-3, at time T2, the RV-server receives a second synchronization request message from a synchronization request function called by a calling process of a computer system M2. As the RV-ID of the meeting point is the same as the previous received synchronization request message (RV3), the RV-server adds a record having the fields of this second synchronization request message in the sub-list. In this second synchronization request message, *RV-total-wait-num* is equal to 2. The RV-server detects that the number of records in the sub-list is equal to the *RV-total-wait-num* of second synchronization request message. A synchronization-attained message comprising the number of records and/or the number of records indicating participation, is provided to the synchronization request function. This synchronization request function returns this number to the calling process corresponding to the second synchronization request message. The calling process is unblocked and reaches its rendezvous.

In Exhibit B-4, at time T3, the RV-server receives a third synchronization request message from a synchronization request function called by a calling process of the computer system M3. As the RV-ID of the meeting point (RV3) is the same as the record of the first and second received synchronization request messages, the RV-server adds a record having the fields of this third synchronization request message in the sub-list. In this third synchronization request message, *RV-total-wait-num* is equal to 3. The RV-server detects that the number of records in the sub-list is equal to the *RV-total-wait-num* of first and third synchronization request messages. A synchronization-attained message comprising the number of records and/or the number of records indicating participation, is provided to the synchronization request functions. The synchronization request functions returns this number to the calling processes corresponding to the first and third synchronization request messages. The calling processes are unblocked and reach their rendezvous.

α

The RV-server may delete a sub-list when the last information message is sent in this sub-list. It indicates that all the records in the sub-list have reached their meeting point or their time-out, and in any cases the calling process do not wait anymore for a synchronization request function return.

A calling process may also request the synchronization request function for special services. Thus, the synchronization request function has a command line to request the RV-server to delete all the current sub-lists, it also has a standard output (*stdout*) command line to request the RV-server to provide a view on a screen of all current sub-lists.

The RV-server may comprise modifiable parameters such as :

p : it indicates the port number on which the RV-server listens to the calling processes of the computer machines,

q : it indicates the number of synchronization request messages waiting for a connection with the RV-server,

v : it indicates the name of a vervose file, its lowest and upper level log,

h : it indicates the help that a user may require.

The invention is not limited to the features hereinabove described.

Most of the detailed description refers to a client-server distributed system. However, this invention may also apply to distributed computer systems using an interaction model other than the client-server model, or between various processes co-existing in a single computer station.

This invention also covers the proposed software code itself, especially when made available on any appropriate computer-readable medium. The expression "computer-readable medium" includes a storage medium such as magnetic or optic, as well as a transmission medium such as a digital or analog signal.

α

Exhibit A

A-1 Example of rendezvous 1

Process 1	Process 2
Instruction	Instruction
Instruction	Instruction
Instruction	Instruction
RV1	Instruction
Instruction	Instruction
Instruction	RV1
Instruction	Instruction

A-2 Synchronization request Function parameters

Parameters of function : rendez_vous

Types	Name of parameters
char	*rv_id*
long	*rv_total_wait_num*
unsigned int	*rv_time_out*
unsigned int	*rv_block*
unsigned int	*participate*
char	*host_name*
long	*port*

Exhibit B

B-1 Synchronization request Message

RV-ID	RV-nbwait	Participant	Blocking
RV3	2	1	1

B-2 Sub-list of synchronization request messages : Time :T1

RV-ID	RV-nbwait	Participant	Blocking
RV3	3	1	1

B-3 Sub-list of synchronization request messages : Time :T2

RV-ID	RV-nbwait	Participant	Blocking
RV3	3	1	1

RV-ID	RV-nbwait	Participant	Blocking
RV3	2	1	1

B-4 Sub-list of synchronization request messages : Time :T3

RV-ID	RV-nbwait	Participant	Blocking
RV3	3	1	1

RV-ID	RV-nbwait	Participant	Blocking
RV3	2	1	1

RV-ID	RV-nbwait	Participant	Blocking
RV3	4	1	1

Claims

1. A management module comprising :

- input code (22-1), capable of receiving a synchronization request message, such a synchronization request message comprising a node meeting identifier and a number of participants,
- table code (22-3), capable of maintaining a record comprising node meeting identifier and number of participants for each received synchronization request message,
- parsing code (22-4), capable of, responsive to a received synchronization request message, checking whether the number of records having the same node meeting identifier (RV-ID) and the number of participants comprised in a record having that node meeting identifier (RV-ID) meets a given condition,
- output code (22-2), responsive to a record meeting the given condition, capable of transmitting a synchronization-attained message.

2. The management module of claim 1, wherein the given condition comprises that the number of records having the same node meeting identifier (RV-ID) is equal to or greater than the number of participants in a record having that node meeting identifier (RV-ID).

3. The management module as claimed in any of the preceding claims, wherein the table code (22-3) is capable of creating a table (24) for maintaining records having the same node meeting identifier (RV-ID).

4. The management module as claimed in any of the preceding claims, wherein, upon the record for a received synchronization request message, the table code (22-3) is capable of counting the number of records having the same node meeting identifier (RV-ID).

5. The management module as claimed in any of the preceding claims, wherein the synchronization request message comprises a participant/non participant indication at the node meeting, and wherein the table code (22-3) is capable of maintaining a record comprising the participant/non participant indication of a received synchronization request message.

6. The management module of claim 5, wherein, upon the record for a received synchronization request message, the table code (22-4) is capable of counting the number of records having the same node meeting identifier (RV-ID) and the participant indication.

7. The management module of claim 6, wherein the synchronization-attained message comprises the number of records having the same node meeting identifier (RV-ID) and the participant indication.

8. The management module as claimed in any of the preceding claims, wherein a synchronization request message is adapted to be sent from a synchronization request function (13-M) on request of a calling process (14-M), said calling process providing parameters.

9. The management module of claim 8, wherein the synchronization request function (13-M) is adapted to establish a connection between the calling process and the management module using provided parameters.

10. The management module of claim 8, wherein the synchronization request function (13-M) is adapted to specify, in each message sent to the management module, provided parameters such as an address adapted to reach the management module and a port identifier.

11. The management module of claim 8, wherein the synchronization request function (13-M) is adapted to operate at least one error control on at least one of the provided parameters and, responsive to a detected error on said parameter, to transmit an error message to the calling process (14-M).

12. The management module as claimed in any of the preceding claims, wherein the synchronization request message further comprises an unblocked/blocked indication.

13. The management module as claimed in any of the preceding claims, wherein the output code (22-2) is further capable of, responsive to a received synchronization request message comprises a blocked indication, transmitting a synchronization-attained message

α

to the synchronization request function (13-M) that is capable of transmitting said synchronization-attained message to the calling process (14-M).

14. A method for managing code execution synchronization, said method comprising the steps of:

a. receiving a synchronization request message, such a synchronization request message comprising a node meeting identifier and a number of participants, (102)

b. maintaining a record of node meeting identifier and number of participants for each received synchronization request message, (106)

c. responsive to a received synchronization request message, checking whether the number of records having the same node meeting identifier and the number of participants in a record having that node meeting identifier meet a given condition, (604)

d. responsive to a record meeting the given condition, transmitting a synchronization-attained message (608).

15. The method of claim 14, wherein the given condition of steps c. and d. comprises that the number of records having the same node meeting identifier is equal to or greater than the number of participants in a record having that node meeting identifier (604).

16. The method as claimed in any of claims 14 and 15, wherein step b. comprises creating a table for maintaining records having the same node meeting identifier (105).

17. The method as claimed in any one of claims 14 to 16, wherein step b. comprises, upon the record for a received synchronization request message, counting the number of records having the same node meeting identifier.

18. The method as claimed in any of claims 14 to 17, wherein the synchronization request message of steps a. and b. comprises a participant/non participant indication at the node meeting, and wherein step b. comprises maintaining a record comprising the participant/non participant indication of a received synchronization request message (106).

ɑ

19. The method of claim 18, wherein step b. comprises, upon the record for a received synchronization request message, counting the number of records having the same node meeting identifier and the participant indication.

20. The method of claim 19, wherein the synchronization-attained message of step d. comprises the number of records having the same node meeting identifier and the participant indication (608).

21. The method as claimed in any of claims 14 to 20, wherein a synchronization request message of steps a. and b. is adapted to be sent from a synchronization request function on request of a calling process, said calling process providing parameters.

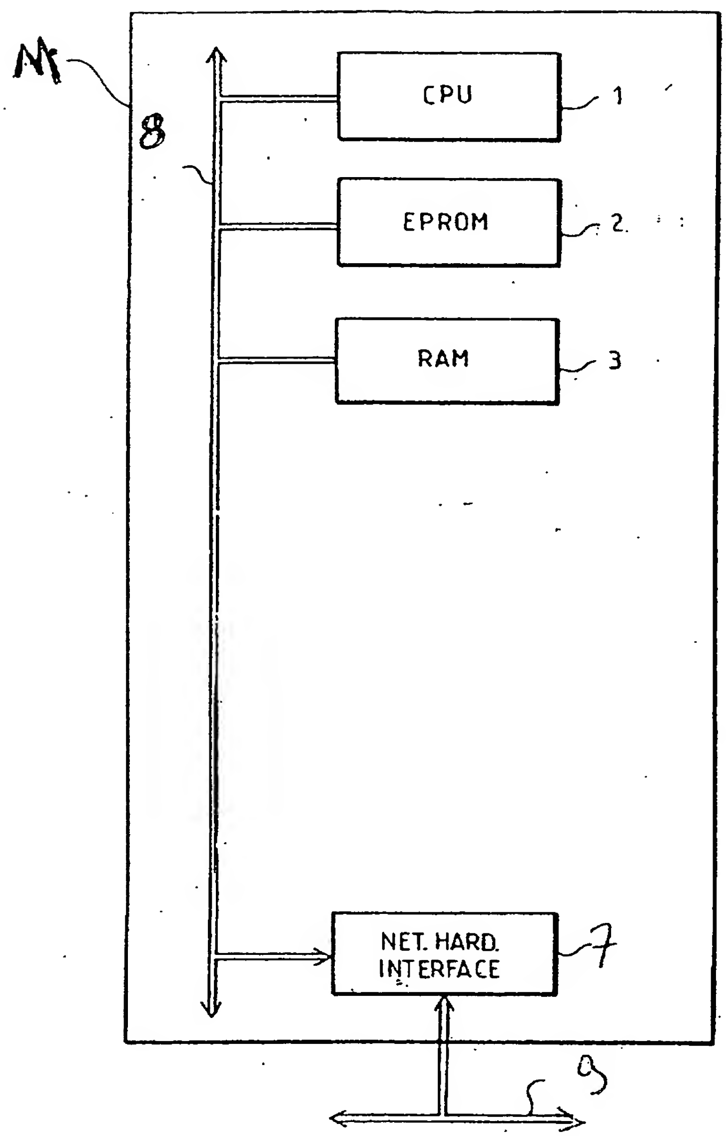
22. The method as claimed in any of claims 14 to 21, wherein the synchronization request function is adapted to operate at least one error control on at least one of the provided parameters and, responsive to a detected error on said parameter, to transmit an error message to the calling process.

23. The method as claimed in any of claims 21 or 22, wherein step d. further comprises, on receipt of a synchronization-attained message, transmitting the return of the synchronization request function to the calling process if the synchronization request message comprises a blocked indication.

24. The method as claimed in any of claims 14 to 23, wherein the synchronization request message of steps a. and b. further comprises an unblocked/blocked indication.

25. The method as claimed in any of claims 14 to 24, wherein step d. further comprises, responsive to a received synchronization request message comprising a blocked indication, transmitting a synchronization-attained message to the synchronization request function that is capable of transmitting said synchronization-attained message to the calling process.

α (11 pages) CABINET NETTER



CABINET NETTER

FIG 1

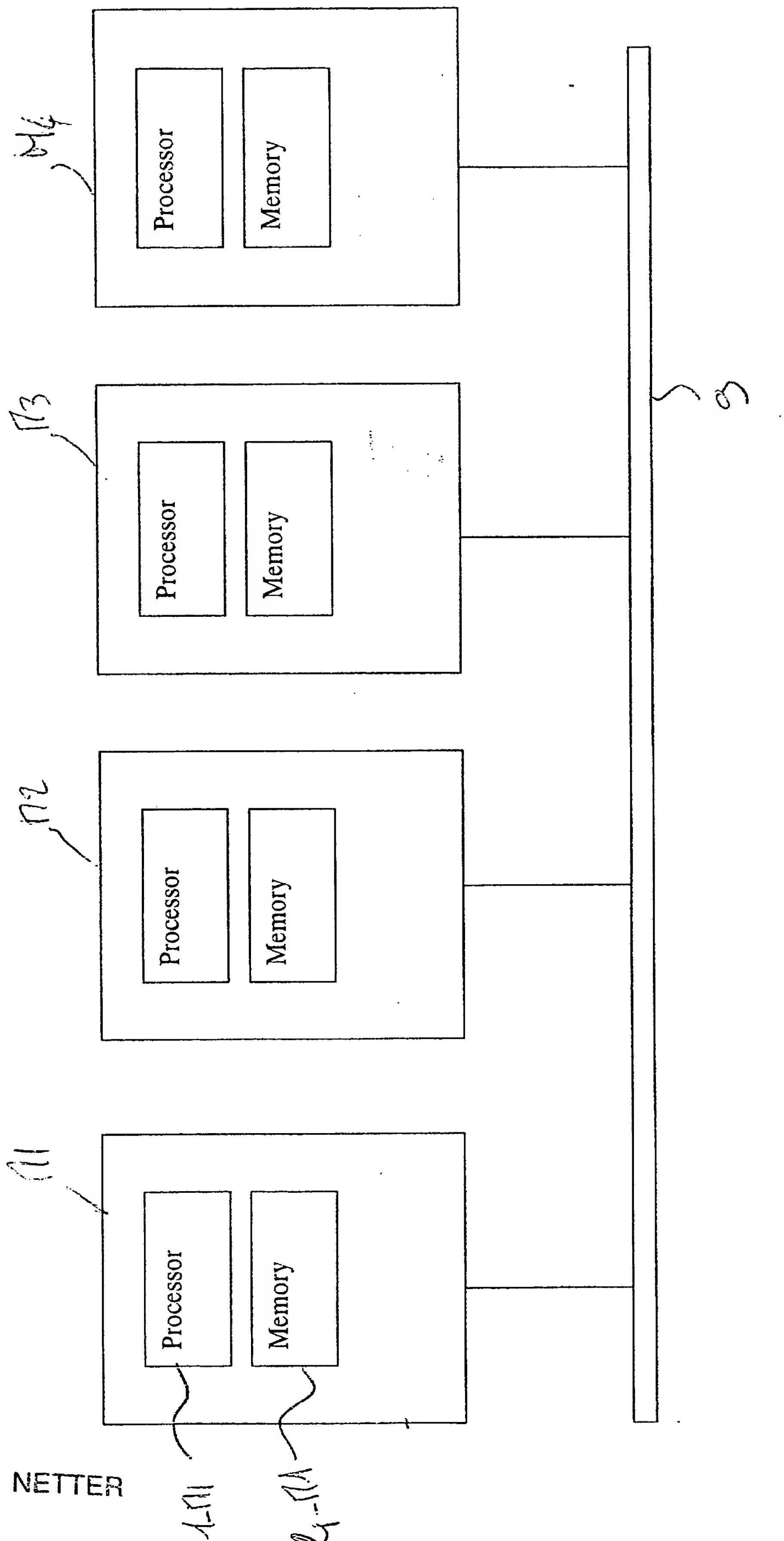

Processor
Memory
Processor
Memory
Processor
Memory
Processor
Memory

Figure 2

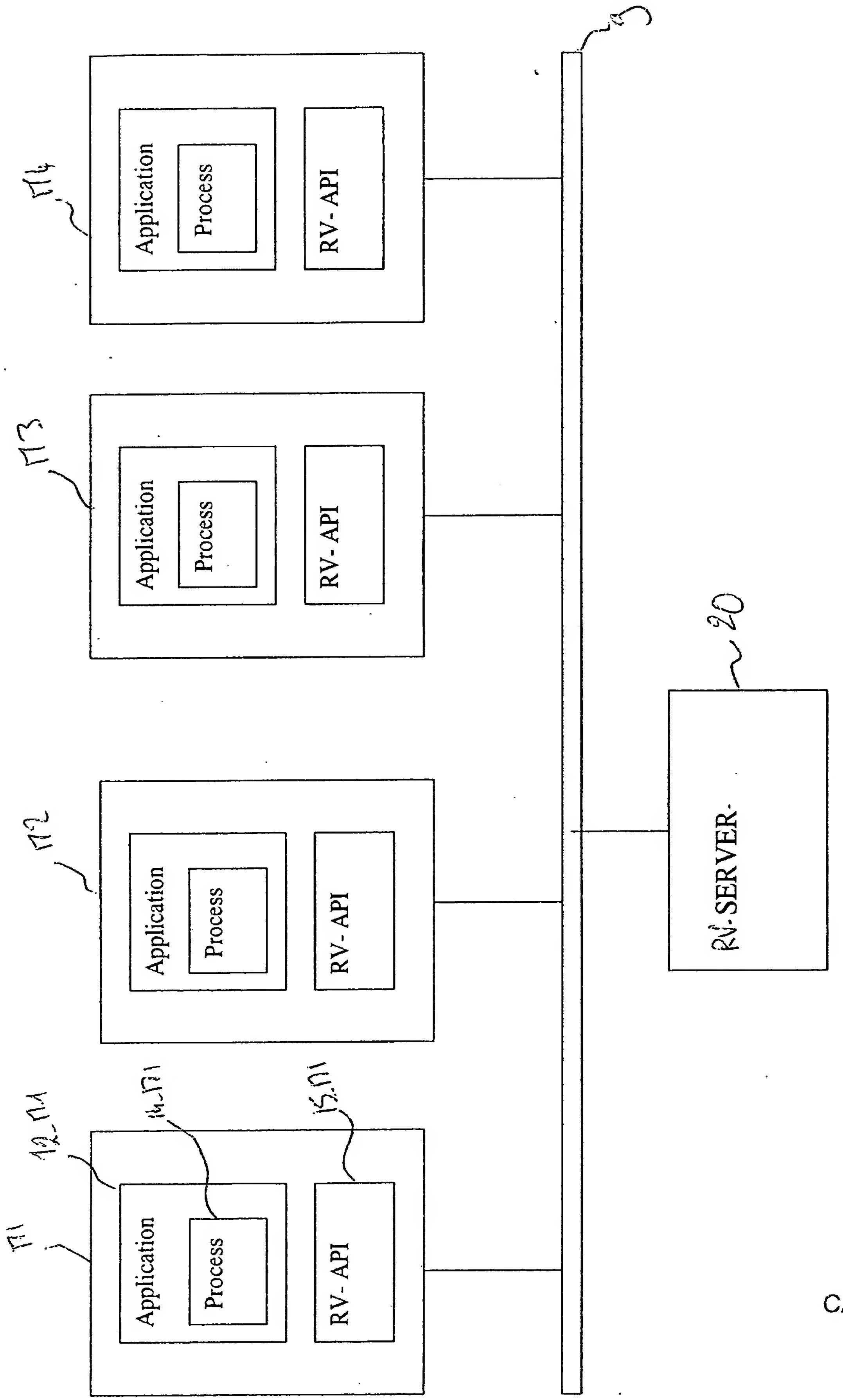


Figure 3

CABINET NETTER

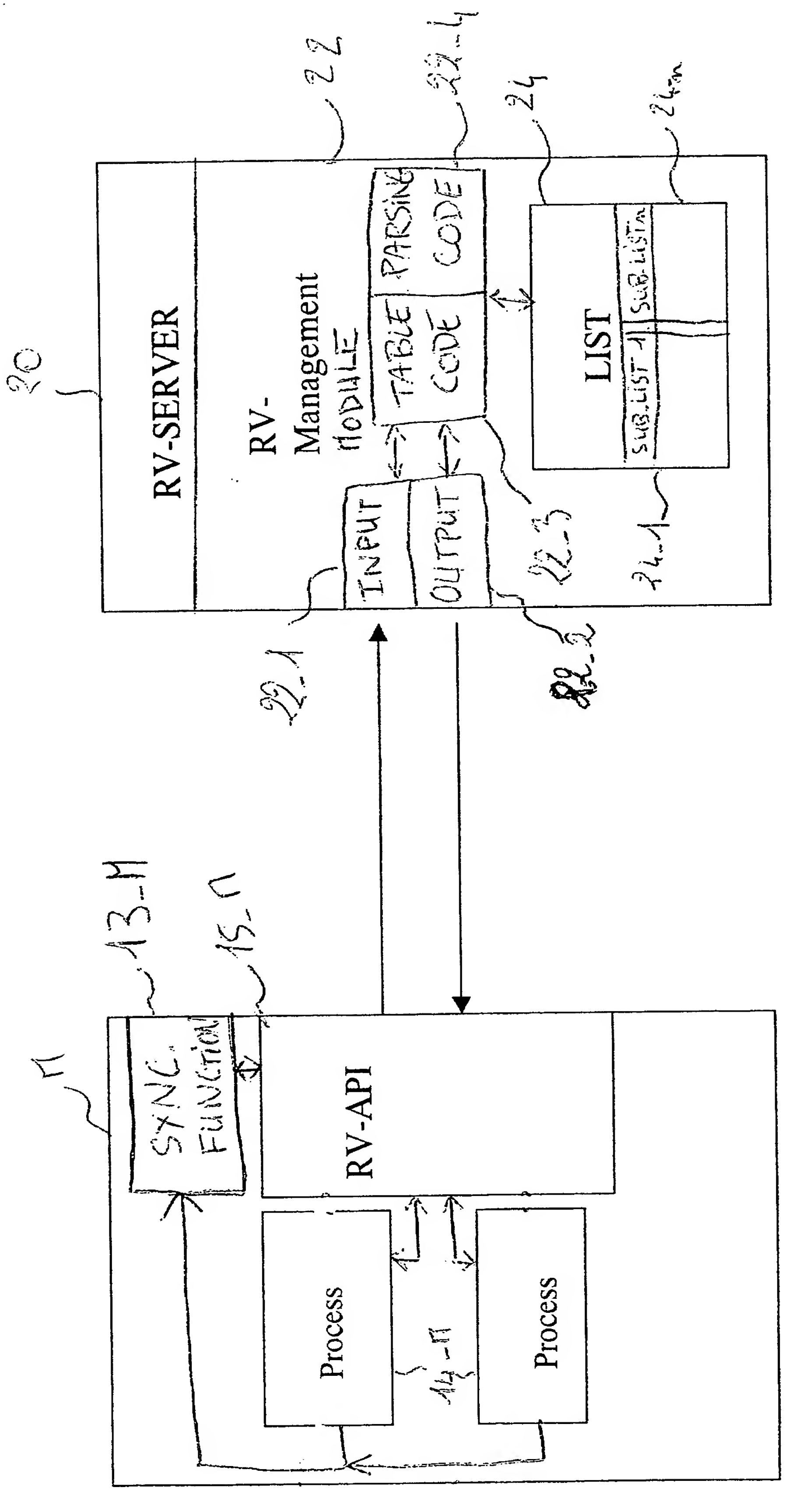
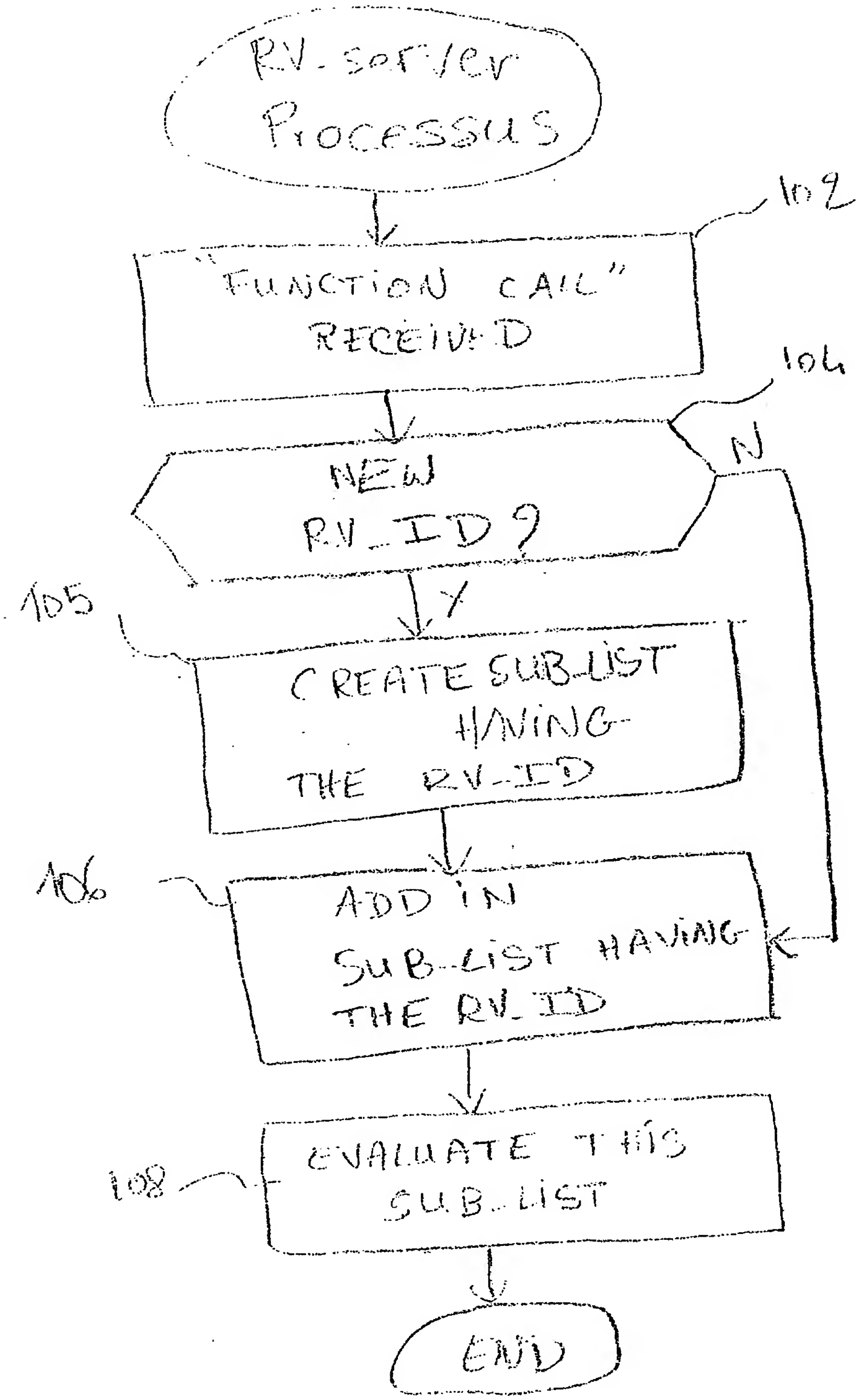


Figure 4

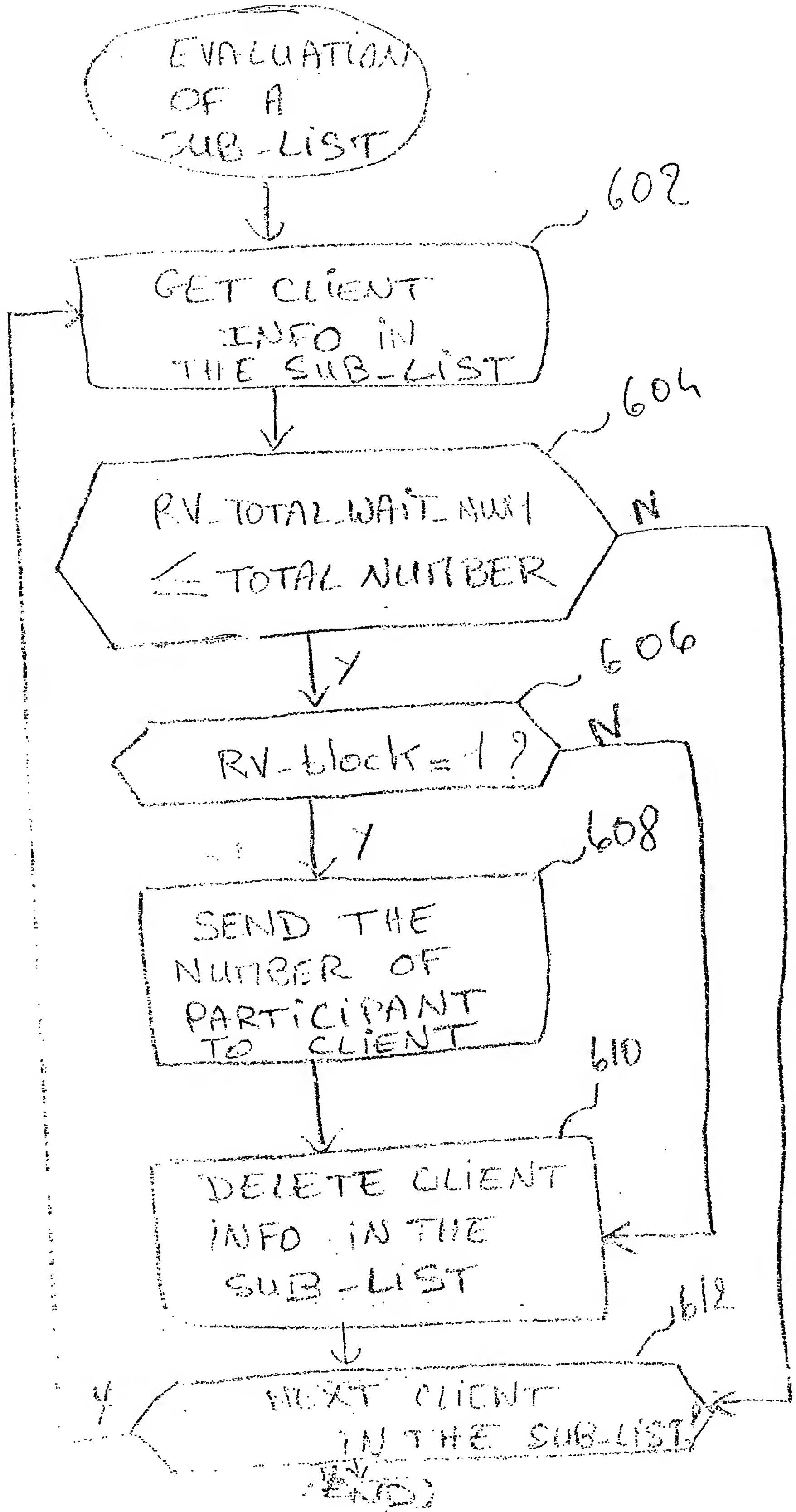
CABINET NETTER



CABINET NETTER

FIG 5

FIG-6



CABINET NETTE.